Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER Computers and Mathematics with Applications 51 (2006) 1705–1714
An International Journal
computers & mathematics
with applications
www.elsevier.com/locate/camwa

A Binary Partitioning Approach to Image Compression Using Weighted Finite Automata for Large Images

GHIM HWEE ONG AND KAI YANG
Department of Computer Science
School of Computing
National University of Singapore
Republic of Singapore
<onggh><yangkai>@comp.nus.edu.sg

Abstract—Fractal-based image compression techniques give efficient decoding time with primitive hardware requirements, and favor real-time communication purposes. One such technique, the weighted finite automata (WFA), is studied on grayscale images. An improved image partitioning technique—the binary or bintree partitioning—is tested on the WFA encoding method. Experimental results show that binary partitioning consistently gives higher compression ratios than the conventional quadtree partitioning method for large images. Moreover, the ability to decode images progressively rendering finer and finer details can be used to display the image over a congested and loss-prone network such as the image transport protocol (ITP) for the Internet, as well as to pave way for multilayered error protection over an often unreliable networking environment. Also, the proposed partitioning approach can be parallelized to reduce its high encoding complexity.

Keywords—Image compression, Weighted finite automata, Quadtree partitioning, Binary partitioning, Data compression.

1. INTRODUCTION

Data compression has been an important issue in relation to transmission and storage of information for a long time. In particular, digital images require a large amount of space in their internal representation often as arrays of pixels, incurring too much storage space and transmission time over a communication channel. Techniques for finding more compact image representations are desired. Also, general-purpose and selectively-reliable transport protocols have been developed to transmit compressed images over the Internet [1].

Many image compression techniques have been developed to suit different applications. Lossy image compression techniques can provide high compression ratios but introduce some annoying artifacts. Fractal-based image compression is one of the lossy data compression techniques that has been developed in the last decade. It makes use of self-similarities existing in an image at

The authors' work was supported by the National University of Singapore Academic Research Fund under Grant R-252-000-118-112.

0898-1221/06/$ - see front matter Typeset by AMS-TEX
doi:10.1016/j.camwa.2006.05.008

different resolutions under a set of affine transformations and exploits this kind of redundancy in order to alleviate the high cost of storage, thus achieving compression.

The weighted finite automata (WFA) method attempts to represent a subimage as a weighted sum of other subimages to accomplish compression. WFA starts with an image to be processed. It locates subimages that are identical or very similar to the entire image or to other subimages, and subsequently constructs a graph that reflects the relationship between these subimages vis-à-vis the entire image. The various components of the graph are compressed and the final product is a much more compactly represented domain.

An important issue involved in fractal-based image compression is the partitioning of an image into subimages or blocks for encoding. There are various approaches to the use of different block shapes. One such commonly used approach is based on quadtree partitioning. It has also been shown that a partitioning scheme based on rectangular blocks is simple yet offers high quality of decomposed images [2]. In this paper, we investigate a simple, new approach to WFA-based image compression with binary partitioning. In particular, we seek to improve the compression ratio by introducing intermediate steps into quadtree partitioning for large images. We will show that binary partitioning gives higher compression than quadtree partitioning for images of 512×512 and 1024×1024 pixels.

In the following sections, the concept of WFA for image compression is first introduced and then a description of the conventional quadtree approach to WFA image encoding is given. They are followed by the description of the proposed binary (or bintree) partitioning approach to WFA coding. Lastly the results of applying both the quadtree and bintree partitioning schemes to large image files are summarized.

2. REVIEW OF WEIGHTED FINITE AUTOMATA FOR IMAGE COMPRESSION

First we introduce WFA as a modelling tool for specifying grayscale images, and then we describe WFA inference algorithms in the next two sections for their construction.

The basic principles of the WFA method are somewhat similar to those of the fractal image compression method based on PIFS (partitioned iterated function systems) [3]. Both the methods employ the fact that images used in practice have a certain amount of self-similarity to achieve compression. In other words, an image portion of the image to be compressed may be similar to another portion of the same image, except perhaps for size, contrast, or brightness. The main difference between the two methods is that the PIFS-based fractal image compression uses affine transformations to match image portions or subimages, while on the other hand WFA describes a subimage as a weighted sum or linear combination of other subimages.

WFA begins with an arbitrary image to be compressed. First the image is split into a set of nonoverlapping image portions or subimages through a given partitioning scheme (e.g., quadtree partitioning as described in the next section). These subimages are identical to those range images used in the PIFS-based fractal image compression method [3]. Any image portion, subimage, or pixel in the image can be denoted by a string of the letters $0, 1, 2, \ldots, d-1$, where d is the least number of range images at a given image resolution. Clearly, a longer string of letters, denoted by $a_1 a_2 \cdots a_k$ (where k is an integer), is used to represent a smaller image portion with higher resolution.

Next, one or more image portions that are very similar or identical to the original image or to each range image to be encoded separately are obtained from a domain pool, and a graph is constructed to describe the relationship between these image portions and the image to be encoded. The domain pool may contain all image portions or subimages which could be generated from the given partitioning scheme. In general, the WFA uses an inference algorithm to construct a graph that is very similar to graphs used to represent finite-state automata [4,5]. The various

components of the graph are then compressed to become the compressed image.

In the graph, one state represents the original image and each of the other states represents a subimage. One or more subimages (or states) can be expressed as weighted sums or linear combinations of other subimages (including the original image) with different weights by means of the edges. For example, subimage i may be expressed as the weighted sum of subimages j and k, respectively, with weights u and v. Then two edges should be constructed, one from state i to state j and another from state i to state k. Thereafter, transition matrices will be derived from the graph. For each transition matrix W_q, element (i,j) of the matrix denoted by $(W_q)_{i,j}$ is either set to the weight u of an edge labeled q from state i to state j, or set to zero otherwise. The label q identifies the subimage in each partition generated by the partitioning scheme at a given resolution. The size of each matrix is derived from the number of states (denoted by m) in the graph and has dimensions $m \times m$. The number of matrices is determined by the partitioning scheme used to generate range images.

Generally, there is an image associated with each state. A column vector F of size m called the final distribution is used to represent all state images. Each element of F is the average intensity (or average grayness) of the subimage associated with one state of the graph. The subimage associated with state i of the graph is represented by ψ_i which can be determined by

$$\psi_i(a_1 a_2 \cdots a_k) = (W_{a_1} \cdot W_{a_2} \cdot \cdots \cdot W_{a_k} \cdot F)_i ,$$

where the string $w = a_1 a_2 \cdots a_k$ specifies the partition that contains the subimage. Note that $\psi_i(w)$ is a number which is the i^{th} element of the column vector $(W_{a_1} \cdot W_{a_2} \cdot \cdots \cdot W_{a_k} \cdot F)^{\mathsf{T}}$, for $i = 1, 2, \ldots, m$. Thus, the pixels of the image of state i can be generated from ψ_i which consists of several numbers for a given value of w. Equivalently, for every state i and $y \in w$, we have

$$\psi_i(qy) = \sum_{j=1}^{m} (W_q)_{i,j} \cdot \psi_j(y).$$

In principle, a multiresolution image A may be represented by an intensity function $f_A(x,y)$ that gives the intensity of the image at point (x,y). WFA defines a row vector with m elements called the initial distribution $I = (I_1, I_2, \ldots, I_m)$. Conceptually, the initial distribution can be used to specify the image defined by a WFA as a linear combination $\sum_{i=1}^{m} I_i \cdot \psi_i$ of state images. Hence, for a given value of $w = a_1 a_2 \cdots a_k$, the intensity function can be expressed as the matrix product

$$f_A(a_1 a_2 \cdots a_k) = I \cdot W_{a_1} \cdot W_{a_2} \cdot \cdots \cdot W_{a_k} \cdot F.$$

For decoding, WFA reads from the compressed stream and uses matrices I, F, and W_q to construct approximately the original image A.

From the above descriptions of the general WFA theory for image compression, we know that in order to arrive at a good approximation of any state image, a linear combination of some known states is employed. It is known that any arbitrary image or image portion can be represented by a linear vector. In mathematical terms, the crucial part of the WFA construction algorithm lies in representing a given vector by linear combinations of other known vectors. To achieve compression gain, it is clear that the algorithm has to approximate a vector with as few others as possible and as accurately as possible. The approximation with matching pursuit (MP) was introduced by Mallet and Zhang [6]. In this technique, an approximation is constructed step by step using a greedy strategy. Essentially, the MP approximation of a vector $\vec{v}$ is constructed by selecting the best matching vector from a given codebook. The component of this best matching vector is then subtracted from $\vec{v}$. The residue is then encoded in the same way as $\vec{v}$ and the process continues until a given abortion criterion is fulfilled.

Let $p, n \in N$ and $D = \{g_0, \ldots, g_{p-1}\}$ with $g_i \in R^n$ $(i \in \{0, \ldots, p-1\})$ be a set of $p \geq n$ nonzero vectors. The set D is often called the dictionary, codebook, or domain pool. With

span (D) we denote the set of all linear combinations of vectors in D. In order to make the following calculations a bit easier we assume that, without loss of generality, each vector in the dictionary has unit Euclidean norm $\|\vec{v}\|$. If span $(D) = R^n$ then D contains a set of n linearly independent vectors. The matching pursuit algorithm begins by projecting $\vec{v}$ on a dictionary vector g_{i_0} and computing the residue Rv by

$$\vec{v} = \langle \vec{v}, g_{i_0} \rangle \, g_{i_0} + Rv.$$

Since Rv is orthogonal to g_{i_0} (by virtue of the vector inner product), the following equation holds:

$$\|\vec{v}\|^2 = |\langle \vec{v}, g_{i_0} \rangle|^2 + \|Rv\|^2.$$

Hence, we have to choose g_{i_0} such that $|\langle \vec{v}, g_{i_0} \rangle|$ is maximized, since residue $\|Rv\|$ has to be minimized. The next iteration continues with Rv instead of $\vec{v}$, and we compute the following iteration.

1. Set $R^0 v = \vec{v}$.
2. Subsequent residues are computed by solving simultaneously

$$R^m v = \langle R^m v, g_{i_m} \rangle \, g_{i_m} + R^{m+1} v,$$
$$\|R^m v\|^2 = |\langle R^m v, g_{i_m} \rangle|^2 + \left\|R^{m+1} v\right\|^2,$$

   and by choosing g_{i_m} to maximize each of the $|\langle R^m v, g_{i_m} \rangle|$.
3. Finally, summing up the last equation in m from 0 to a stopping index $M - 1$ yields

$$\vec{v} = \sum_{m=0}^{M-1} \langle R^m v, g_{i_m} \rangle \, g_{i_m} + R^M v.$$

3. THE QUADTREE APPROACH TO WFA IMAGE ENCODING

Quadtree partitioning is a method used to represent an image as a hierarchical quadtree data structure in which the root of the tree is the initial image and each node contains four subnodes. A node represents a square image portion and its four subnodes correspond to the four quadrants of the square image portion. In general, a multiresolution image is specified by assigning the grayscale value 0 to 255 to every node of the infinite quadtree. If the four outgoing edges of each node of the quadtree are labeled letters 0, 1, 2, and 3, respectively, we get a uniquely labeled path to every node. The label is called the address of the node. The address of a node at depth k is a string of length k letters over the alphabet $\sum = \{0, 1, 2, 3\}$. If the size of the original image is $2^m \times 2^m$ (for $m \in \mathbb{Z}^+$), then the string $a_1 a_2 \cdots a_i \cdots a_k$ or k letters (where $a_i = 0$, 1, 2, or 3) defines a square image portion or subsquare of size $2^{m-k} \times 2^{m-k}$ pixels. Hence, a grayscale multiresolution image can be specified as a subset of strings over the alphabet $\sum$. Regular sets of strings are specified by finite automata; therefore, finite automata can be used to specify regular multiresolution images.

The WFA accepts grayscale pictures containing $2^m \times 2^m$ pixels where pixel intensity ranges from 0 to 255. The image portions or subimages used by WFA are obtained by the quadtree partitioning of the image. If a given image is represented by a WFA, then each state of the WFA must correspond to a subsquare of the image, with the initial state corresponding to the entire image. Moreover, if there is a transition from state i to state j labeled by 0 (or 1, 2, or 3), then the image corresponding to state j is the SW (or NW or SE or NE, respectively) quadrant of the

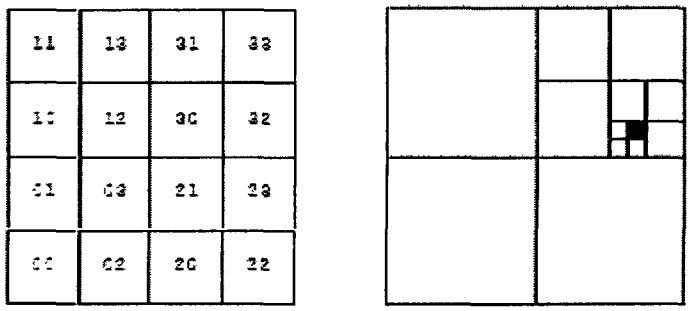


Figure 1. The addresses of the quadrants of the 4 × 4 square, and the subsquare specified by 3203.

image corresponding to state i. An example as given in Figure 1 shows the recursive zooming of the quadtree.

According to Culik and Kari [4,5], a WFA specifying an image does not merely search for one of the known states that best matches the substate under investigation; instead, it uses a linear combination of (possibly all) known states to arrive at a better approximation. In general, an m-state WFA A, over alphabet $\sum$, is specified by

1. A row vector $I \in R^{1\times m}$ (called initial distribution).
2. A column vector $F \in R^{m\times 1}$ (called final distribution).
3. Weight transition matrices $W_a \in R^{m\times m}$ for all $a \in \sum$.

To display the WFA using a graph, we represent the m states by circles with the initial (first value) and final (second value) distribution shown inside each state. If $(W_a)_{i,j} \neq 0$, place an edge from state i to state j, labeled by $a((W_a)_{i,j})$. For example, suppose that we have

$$\text{the initial distribution} \qquad I = (1 \quad 0);$$

$$\text{the final distribution} \qquad F = \begin{pmatrix} 1 \\ \frac{1}{2} \\ 1 \end{pmatrix};$$

$$\text{the weight transition matrices} \quad W_0 = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{4} \\ 0 & 1 \end{pmatrix}, \quad W_1 = \begin{pmatrix} 1 & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix},$$

$$W_2 = \begin{pmatrix} 1 & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}, \quad W_3 = \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} \\ 0 & 1 \end{pmatrix}.$$

Each state represents a portion of the image, and the edges attached to a state represent the linear combination of states which, when calculated, produce an approximation of the image portion. The graph of such a WFA is illustrated in Figure 2.

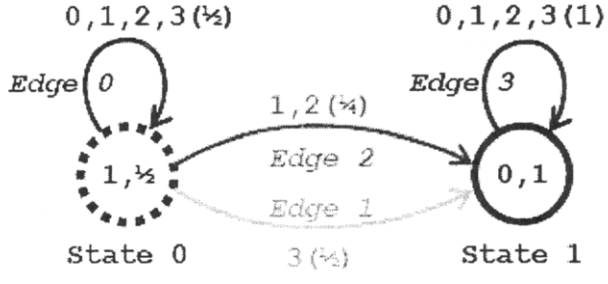
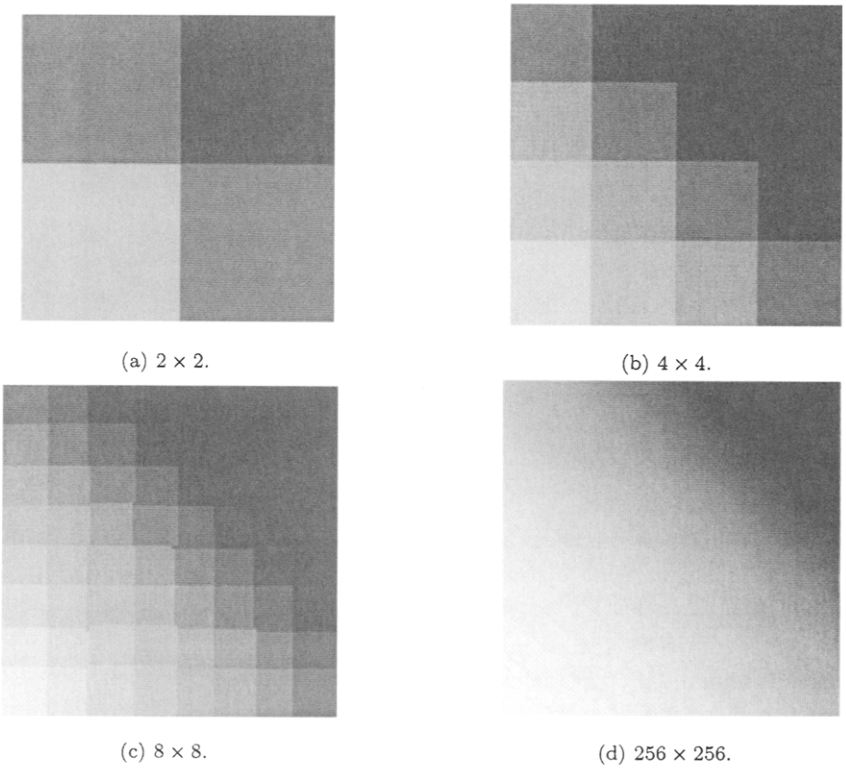


Figure 2. A graph representation of the WFA.



(a) 2 × 2. (b) 4 × 4.

(c) 8 × 8. (d) 256 × 256.

Figure 3. The image at resolution described by the WFA given in Figure 2.

Given the WFA A over alphabet $\sum$ shown in Figure 2, the initial distribution $I = (1 \quad 0)$ is obtained from the first value inside each state, ordered by the state number; similarly the final distribution

$$F = \begin{pmatrix} \frac{1}{2} \\ 1 \end{pmatrix}$$

is obtained from the second value inside each state.

When an edge exists from state i to state j (i and j may be the same state), place the value enclosed in parenthesis in position (i, j) of matrix W_a where a is the label associated with the edge. For example, the bottom edge joining state 0 to state 1 has label "3"and a value of 1/2; in position $(0, 1)$ of weight matrix W_3 we place the value 1/2. The weight of a path is calculated by multiplying the weights of all the edges on the path, the initial distribution value from the first state on the path, and the final distribution value from the last state on the path. To determine the value of the pixel represented by quadrant "03", we calculate: $I \times W_0 \times W_3 \times F = 3/8$. Using the given WFA A, the actual image for resolutions 2 × 2, 4 × 4, 8 × 8, and 256 × 256 is shown in Figure 3.

4. THE PROPOSED BINARY OR BINTREE APPROACH TO WFA IMAGE ENCODING

The discussion so far has been under the assumption that the WFA is approximated by recursively dividing into four image quadrants, each zooming to a higher level of detail. However, each quadtree partition looks at only 25% effective area of the original state, and much of the spatial correlations are lost in the division process.

Seeing in this light, we propose a modification of WFA image approximation using binary partitioning. The idea is to first divide the current (square) state q horizontally into two (rectangular) substates q'_0 and q'_1, and later divide q'_0 and q'_1 again vertically to produce four (square)

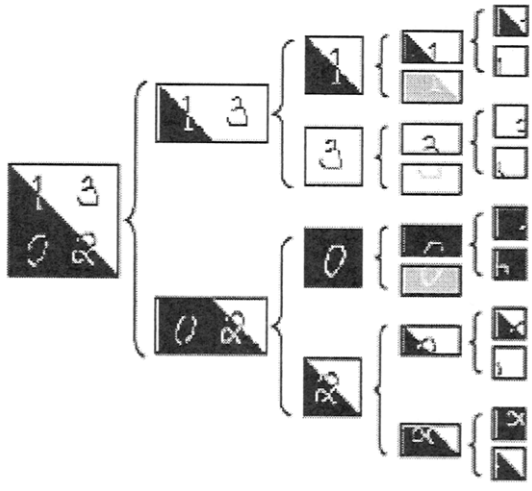


Figure 4. The image is divided twice: horizontally followed by vertically (redundant subimages are shown in pale shades).

substates q''_{00}, q''_{01}, q''_{10}, and q''_{11} just as what a single division of the quadtree partitioning would have done. This effectively separates the previous quadtree partitioning into two steps: a horizontal cut followed by a vertical cut. The rational behind this separation is to better utilize the spatial correlations among image segments, by looking at 50% effective area. An example is given below in Figure 4. Instead of mechanically following the horizontal and then vertical cut to every (square) state image, a further enhancement tests both alternatives and picks the better choice. However, this involves some back-tracking.

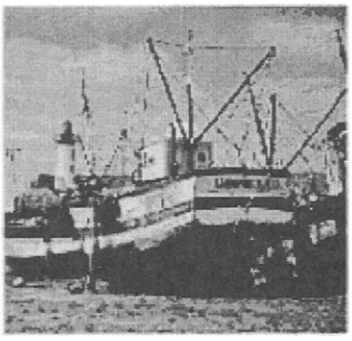
An outline of the WFA construction algorithm is presented below: suppose that an image has been partitioned into a set of nonoverlapping regions. Starting with the complete image, the current range image is recursively inspected if it can be approximated with a linear combination of arbitrary known subimages (using methods such as MP discussed earlier). If a linear combination yields only a poor approximation, this range image will be divided with bintree partitioning again and the recursion continues, while a new state image is appended to the WFA under construction. On the other hand, if this approximation already satisfies the given quality threshold, the corresponding transitions are appended to the WFA and the recursion terminates. During compression, the domain pool grows dynamically: at the beginning the domain pool consists only of some initial basis (such as polynomial functions); during runtime every successfully approximated range image will be added to the domain pool. This means that the more range images are used for image partitioning, the more domain images will be available during later approximation. Consequently, the finer the partitioning of the image, the better would be the image quality.

The storage of automata uses two bits for each quadrant (quadtree) or one bit for each half (bintree) of a tree node. This defines whether the segment was approximated by matching pursuit vectors or whether it points to a new node. In the former case, we have to subsequently encode the MP vectors, while in the latter case, we just mark the index of the node to which this node points. In order to determine the end of the tree structure, we store the total number of tree nodes at the very beginning. The entire tree structure sequence is stored using an adaptive arithmetic coder [7] since the probability of a split is higher near the root of the tree.

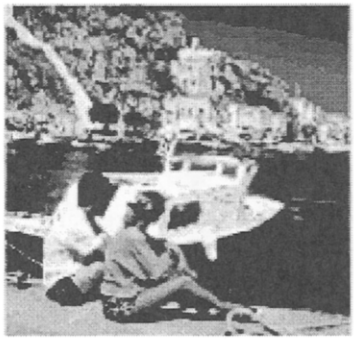
The main task of the image regeneration involves calculation of all the range images. This only requires some simple manipulation for each range image

$$\psi = \sum_{i=1}^{m} w_i \cdot W_i,$$

Lena.



Boat.



Couple.

Peppers.

Figure 5. Four images used in the test.

where w_i are the (quantized) weights and W_i are the weight transition matrices for automaton in the linear combination of m domain images. Every range image must not only be computed at its fixed size in the original image but also at any desired size in the linear combinations. Moreover, domain images that are referred to more than once are cached (stored in an array) to avoid multiple computations of the same image segment.

5. RESULTS AND DISCUSSIONS

Four grayscale images (shown in Figure 5), which are commonly used in the community of image processing and compression, were used to provide the image sample data with which to experiment in this paper. Tests were then conducted on these images of two sizes, 512×512 and 1024×1024 pixels, using binary partitioning and the quadtree partitioning methods. The evaluation of the proposed partitioning method is made on the basis of the results of compression, image quality, and CPU processing times on the four images.

The compression ratio is defined as the ratio of the number of bits in the original image file to the number of bits in the compressed file. The quality of images is determined by the peak signal-to-noise ratio (PSNR), which determines the difference between two images. It is defined as

$$\text{PSNR} = 20 \log_{10} \left(\frac{b}{\text{rms}} \right) dB,$$

where b is the largest possible value of the signal or pixel value, and rms is the root-mean-square difference between two images. The PSNR is given in units of decibel (dB), which measure the ratio of the peak signal and the difference between two images. That is to say, the higher the value, the better the image quality.

Experimentally, it is found that the compression ratios are nearly independent of the initial vertical or horizontal rectangles chosen for binary partitioning. For each image size, the com-

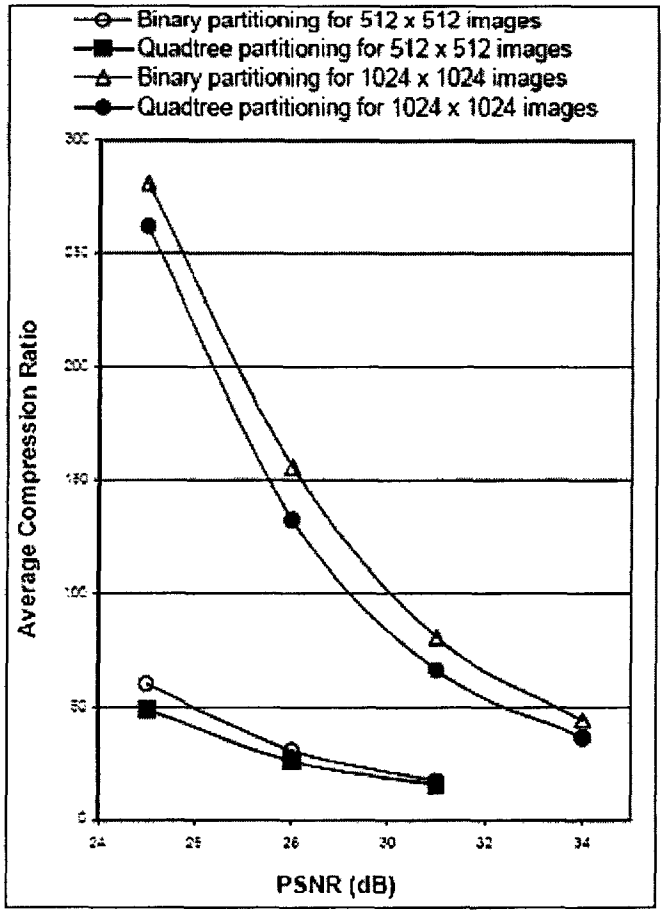


Figure 6. Average compression ratio versus PSNR results for the four images.

pression results are measured with both the binary and quadtree partitioning methods for PSNR values ranging from 25 to 34. Figure 6 depicts the average compression ratio versus PSNR results for the four images.

It is observed that the binary partitioning method consistently gives better compression than the quadtree method. This confirms that there are good matches for the rectangular blocks and the encoding of these blocks (instead of splitting into squares) gives better compression. The results show that the improvement in the compression ratio increases with the size of the image. On average, the compression ratio improves from about 7 to 22%.

Experimentally, it is observed that the binary partitioning method takes a little longer to encode than the quadtree method. This is due to the additional steps taken by binary partitioning to search for some best matches. The decoding times for both the methods are found to be comparable. However, the proposed binary approach to WFA is of high encoding complexity. This is because a domain pool has to be generated to contain one or more image portions as initial bases for encoding large images. The pool can be created by means of a full search scheme or a two-level fast search method based on the distribution of best matched domains. The full search scheme exhaustively compares all entities in the domain pool with a given range block when searching for the best domain. As for the two-level fast search method, the first level of searching previews various areas of an image and serves as a filter to classify promising domains from poor domains. The second level of searching picks up domain blocks in the area where the first level gives positive results, and compares them with a given range block. To further reduce the encoding times, it is possible to employ a parallel full search algorithm using both

static domain distribution and static range distribution schemes [8], or a parallel two-level search algorithm using a dynamic range distribution scheme to achieve a speedup of about 10 with 13 processors [9].

6. CONCLUSION

In this paper, we have presented a binary partitioning method for the WFA-based image compression. A square image is split into rectangular and square blocks with binary partitioning of the image. Experimental results show that binary partitioning gives higher compression results than quadtree partitioning for large images. The improvement can be as high as about 22% for 1024×1024 images. From the transmission perspective, the proposed binary partitioning approach to WFA-based image compression can be designed to decode partially received, out-of-order image data. This scheme can be customized to couple with the image transport protocol for image transmission over loss-prone congested or wireless networks. Also, the proposed approach can be parallelized to reduce its high encoding complexity.

REFERENCES

1. S. Raman, H. Balakrishnan and M. Srinivasan, An image transport protocol for the Internet, In *Proceedings of International Conference on Network Protocols*, pp. 209-219, Japan, (2000).
2. N. Ponomarenko *et al.*, Modified horizontal vertical partition scheme for fractal image compression, In 5^{th} *Nordic Signal Processing Symposium*, `http://www.norsig.no/norsig2002`, Norway, (2002).
3. Y. Fisher, Editor, *Fractal Image Compression—Theory and Applications*, Springer-Verlag, New York, (1995).
4. K. Culik II and J. Kari, Image compression using weighted finite automata, *Computer and Graphics* **17**, 305–313, (1993).
5. K. Culik II and J. Kari, Efficient inference algorithm for weighted finite automata, In *Fractal Image Compression*, (Edited by Y. Fisher), Springer-Verlag, New York, (1994).
6. S.G. Mallet and Z. Zhang, Matching pursuits with time-frequency dictionaries, *IEEE Transactions on Signal Processing* **41** (12), 3397–3415, (1993).
7. I.H. Witten, R.M. Neal and J.G. Cleary, Arithmetic coding for data compression, *Comm. ACM* **30** (6), 520–540, (1987).
8. G.H. Ong and L. Fan, Parallel processing for fractal image compression based on full searching and hexagonal partitioning, In *Proceedings of 2001 International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, pp. 61–66, Wuhan, China, (2001).
9. G.H. Ong and L. Fan, An efficient parallel algorithm for hexagonal-based fractal image compression, In *Proceedings of Joint International Symposium on Distributed Computing and Applications to Business, Engineering and Science and International Conference on Parallel Algorithms and Computing Environments*, London, United Kingdom, 2005, pp. 139–142.